LOGIC PROGRAMMING AS DIALOG-GAME

Phan Minh Dung
Division of Computer Science
Asian Institute of Technology
GPO Box 2754, Bangkok 10501, Thailand
E-mail:dung@cs.ait.ac.th

Abstract

This paper presents a game-theoretical semantics for logic programs, a novel and radically new approach to semantics of logic programming. Two results are obtained:

The first is a novel and simple theory of abstract dialog-games.

In the second result, we show that a game-theoretical semantics for logic programming can be given by interpreting logic programs as schema for forming arguments. According to this game semantics, a proposition can be concluded from a logic program if it is supported by an acceptable argument constructed according to the rules encoded as clauses in the logic program. Depending on the structure of arguments, our game-semantics correponds either to the Fitting's semantics or to the Well-founded semantics of logic programming.

Our result, to the best of our knowledge, is the first ever known result which points out the intimate relationship between logic programming and game-theoretical semantics, two fields which seem, until now, totally unrelated.

Keywords: Logic Programming, Semantics, Dialog-Games

"The true basis of the logic of existence and universality lies in the human activities of seeking and finding"

Jaakko Hintikka [H,pp33]

0. Introduction

In the last three decades, much works have been done in philosophy and logic to study the relationship between the way humans argue in their daily lives and the semantics of (monotonic) logic and natural language [H,BM,L]. These works represent a radical and couragous deviation from the traditional Tarskian semantics. The basic idea here goes back to Kant who held, according to Hintikka [H], "that our ways of reasoning about existence, especially about interindividual existential inferences, must be grounded in human activities through which we come to know the existence of individuals". Nowadays this idea is shared by a number of philosophen and logicians [BM,H,L]. Lorenz and Lorenzen [L,BM] are among the first who have given much life to this idea by showing that classical first order logics can be viewed as dialog-game logics where propositions are entities which can, according to the rules of a dialog-game, either won or lost. Another forceful proponent of this idea is Hintikka who argues convincingly in his numerous works [H] that the semantics of logic and natural language should be based on a game-theoretical basis. For Hintikka, the true basis of the logic of existence and universality lies in the human activities of seeking and finding [H].

Logic programming is widely accepted today as the most suitable programming environtment for developing intelligent systems which are capable of performing "human-like" reasoning. The reason for this, we argue, lies in the fact that the semantics of logic programming, as it will be showed in this paper, resemble very much the way humans argue, the way humans reason through dialog. In fact, we will show that

> "The semantics of logic programming can be based on a sound and intuitive game-theoretical basis"

A dialog-game is an exchange of arguments between two players in which each of them alternately presents an argument attacking the previous one of the opponent. The player who fails to present such an argument in his turn loses the game. A game-theoretical semantics for logic programming can be given by interpreting logic programs as schema for forming arguments. A proposition can be concluded from a logic program if it is supported by an acceptable argument constructed according to the rules encoded as clauses in the logic program. Let us illustrate this by an example.

Example (Why Tweety doesn't fly)

Let P be the logic program:

```
fly(X) <- bird(X), ¬ab(X)
ab(X) <- penguin(X)
penguin(Tweety)
```

P is viewed as the rules for constructing the following two arguments:

A1: Tweety flies since he is a bird and birds normally fly.
A2: Tweety is a abnormal bird since it is a penguin and penguins are abnormal birds

A dialog-game to establish whether Tweety can fly proceeds as follows: Player1 starts by presenting argument A1 supporting the conclusion that Tweety can fly. The argument of player1 is based on a (default) assumption which says that birds normally fly. In the next move, player2 presents A2 which attacks A1 by attacking the default assumption of A1. As player1 can not find any argument to defeat this attack, he loses the game. So he has to give up his claim. □

The purpose of this paper is to present a novel, radically new, game-theoretical semantics for logic programming. Two results are obtained: First, a novel and simple theory of abstract dialog-games is given. Second, we show that a game-theoretical semantics for logic programming can be given by interpreting logic programs as schema for forming arguments. Depending on the structure of arguments, which are constructed according to the rules encoded as clauses in the logic program, our game-semantics correponds either to the Fitting's semantics or to the Well-founded semantics of logic programming.

Our result, to the best of our knowledge, is the first ever known result which points out the intimate relationship between logic programming and game-theoretical semantics, two fields which seem, until now, totally unrelated.

The paper is organized as follows: In chapter 1, we introduce the concept of abstract dialog-games. Then in chapter 2, a fixpoint theory of game semantics is given. In chapter three, we show how the Fitting's semantics of logic programming can be interpreted as dialog game semantics. Chapter 4 shows that logic programming with well-founded semantics can also be viewed as dialog game.

1. Abstract Dialog-Games

A dialog-game is an exchange of arguments between two players in which each of them alternately presents an argument attacking the previous one of the opponent. The player who fails to present such an argument in his turn loses the game, i.e. the other player wins.

Our theory of dialog games is based on the notion of argumentation frameworks.

Definition

An argumentation framework is a pair

$$AF = (AR, attacks)$$

where AR is a set of possible arguments, and attacks is a binary relation on AR, i.e. attacks $\subseteq$ AR x AR. □

Remark 1 (A,B) $\in$ attacks means that A is a counterargument of B, or in other words, A represents an attack on B. We also often say that A attacks B (or B is attacked by A) if attacks(A,B) holds.

Remark 2 From now on, if not explicitly mentioned otherwise, we always refer to an arbitrary but fixed argumentation framework AF = (AR,attacks).

Remark 3 For each argument A, the set of all arguments which attack A is denoted by at(A), i.e. at(A) = { B ¦ (B,A) $\in$ attacks }.

Now, we can introduce our definition of games which is a simple generalization of Hintikka's games for first order logic [H].

Let A be an argument. A game to defend A, denoted as GG(A), between a proponent P and an opponent O, is defined as follows:

> The game starts with the opponent O putting forward an argument B which attacks A. If O fails to present such an argument, then the game stops with a win for the proponent P. Otherwise, the game continues as a game GG(B) with interchanged roles for the players, i.e. O as the proponent and P as the opponent in GG(B).

We say that an argument A is acceptable if there exists a winning strategy in GG(A) for the proponent, i.e. a way for the proponent to choose his moves such that he ends up winning no matter what the opponent does.

A formalization of the above informal definitions of game and acceptable arguments will be given now.

A possible game of length n to defend A is a sequence $A=A_0, A_1,\ldots,A_n$ such that for each $n>i\geq 0$, A_{i+1} attacks A_i.

Let **GAM(A)** be the set of all possible games to defend A. Further, let $\mathbf{GAM_0(A)}$ (res. $\mathbf{GAM_1(A)}$) be the set of possible games of even (resp. odd) length in GAM(A).[1]

Definition

A strategy for the proponent in games to defend A is a partial mapping

$$st: GAM_1(A) \rightarrow AR$$

such that for each $G = (A_0,\ldots, A_{2n+1}) \in GAM_1(A)$, if st(G) is defined, then $st(G) \in at(A_{2n+1})$.

A strategy for the opponent in games to defend A is a partial mapping

$$st: GAM_0(A) \rightarrow AR$$

such that for each $G = (A_0,\ldots, A_{2n}) \in GAM_0(A)$, if st(G) is defined, then $st(G) \in at(A_{2n})$. □

Definition

A game of length n to defend A wrt the strategies st_P, st_O for the proponent and opponent respectively, is a sequence $G = (A_0,A_1,\ldots, A_n\ldots)$ satisfying the following properties:

1) $A = A_0$

2) for each $i\geq 0$, if A_{2i} is not the last element in G then $st_O(G_{2i})$ is defined and $A_{2i+1} = st_O(G_{2i})$

---

[1].That means that GAM_0 (resp. $GAM_1(A)$) is the set of possible games to defend A where the next move is an opponent's (resp. proponent's) move.

3) for each i>0, if A_{2i-1} is not the last element in G then $st_p(G_{2i-1})$ is defined and $A_{2i} = st_o(G_{2i-1})$

4) If G is finite with A_n as the last element then

4.1) $st_o(G_n)$ is undefined if n is even, and

4.2) $st_p(G_n)$ is undefined if n is odd.

where $G_i = (A_0,\ldots,A_i)$. □

The set of all games to defend A wrt the strategies st_p, st_o for the proponent and opponent, respectively, is denoted by $GG(A,st_p,st_o)$.

A finite game of even length in $GG(A,st_p,st_o)$ is a successful game (for the proponent). A game is a lost game (for the proponent) if it is not successful.

Definition

A strategy st_p is called a winning strategy for A iff for each strategy st_o of the opponent, all games in $GG(A,st_p,st_o)$ are successful. □

Definition

An argument A is acceptable iff there exists a winning strategy for A. □

The set of all acceptable arguments of AF is denoted by $\mathbf{ACC}_{AF}$.

2. A Fixpoint Semantics for Abstract Dialog-Games

First let us prove two simple lemmas.

Lemma 1

Let st_p be a winning strategy for A. Further let $B \in at(A)$. Then $st_p((A,B))$ is defined, and st_p is also a winning strategy for $A' = st_p((A,B))$. □

We say that an argument A is defended by a set of arguments S if for each argument B if B attacks A then B is attacked by an argument in S.

It follows immediately from the above lemma that if A is acceptable then A is defended by a set of acceptable arguments. It holds even more:

Lemma 2

An argument A is acceptable iff it is defended by ACC_{AF}.

Proof "=>" Obvious.
"<=" For each $B_i \in at(A) = \{B_1,..,B_i,..\}$, let A_i denote an arbitrary but fixed element in $at(B_i) \cap ACC_{AF}$. Let st_i denote a winning strategy of A_i. Define a strategy st as follows: st is a partial mapping from $GAM_1(A)$ into AR s.t. for each $G = A,B_i,G'$, $st(G) = A_i$, if G' is empty, otherwise $st(G) = st_i(G')$. We want to prove now that st is a winning strategy for A. Assume the contrary. Then there is a strategy st_0 for the opponent such that $GG(A,st,st_0)$ contains a lost game $G0 = (A=C_0,..,C_{2n+1},...)$. Let $C_1 = B_i$. Then it is clear that $C_2 = A_i$. Therefore $C_2,..C_{2n+1}...$ is a lost game for A_i wrt winning strategy st_i. Contradiction. □

This motivates the definition of the following operator:

Define $F_{AF}: 2^{AR} \rightarrow 2^{AR}$ by

$F_{AF}(S) = \{ A \mid A \text{ is defended by } S \}$

Remark As we always refer to an arbitrary but fixed argumentation framework AF, we often write F shortly for F_{AF}.

It is easy to see that if an argument A is defended by a set of arguments S then A is also defended by any superset of S. Thus

Lemma 3

F_{AF} is monotonic (wrt set inclusion). □

We want to show now that the set of all acceptable arguments form the least fixpoint of F.

Theorem 1

An argument is acceptable iff it belongs to the least fixed point of F_{AF}.

Proof "<=" Obvious from the above lemma.
"=>" Assume the contrary. Let A be an acceptable argument not

belonging to lfp(F). Let st_p be a winning strategy for A. Let st_o be a strategy for the opponent defined as follows: for each $G \in GAM_o(A)$, $st_o(G) = A'$ s.t. $at(A') \cap lpf(F) = \emptyset$ where B is the last elemnt of G and $A' \in at(B)$. It is not difficult to see that if $st_o(G)$ is not defined then $B \in lfp(F)$. Since st_p is a winning strategy for A, there exists a successful game $A_1,...A_n$ in $GG(A,st_p,st_o)$. Therefore, n=2i and $st_o(A_n)$ is not defined. Thus $A_n \in lfp(F)$. That means that $at(A_{n-1}) \cap lfp(F) \neq \emptyset$. Thus, $A_{n-1} \neq st_o(A_{n-2})$. Hence the game $A_1,..,A_n$ is not in $GG(A,st_p,st_o)$. Contradiction. □

This theorem is very important as it gives us practically a method to compute all the acceptable arguments. But unfortunately, F_{AF} is not continuous, in general. But if the argumentation framework is finitary then it is.

Definition An argumentation framework AF = (AR,attacks) is finitary iff for each argument A, there are only finitely many arguments in AR which attack A. □

Lemma 4

If AF is finitary then F_{AF} is ω-continuous.

Proof Let $S_0 \subseteq ... \subseteq S_n \subseteq ..$ be an increasing sequence of sets of arguments, and let $S = S_0 \cup ... \cup S_n \cup ...$ Let $A \in F_{AF}(S)$. Since there are only finitely many arguments which attack A, there exists a number m s.t. $A \in F_{AF}(S_m)$. Therefore, $F_{AF}(S) = F_{AF}(S_0) \cup ... \cup F_{AF}(S_n) \cup ....$ □

Corollary In an finitary argumentation framework, an argument A is acceptable if and only if it belongs to $F^i(\emptyset)$ for some natural number i. □

3. Logic Programming with Fitting's Semantics as Games

The semantics of logic programming with negation as finite failure is Fitting's semantics [F]. In this chapter, a game-theoretical interpretation of Fitting's semantics is given.

A logic program is a set of clauses of the form

$$a_0 \leftarrow a_1,..,a_m,\neg a_{m+1},..,\neg a_{m+n}$$

where a_i's are atoms.

For a logic program P, G_P denotes the set of all ground instances of clauses in P. The Herbrand base of P is denoted by HB_P, or just HB if it is clear from the context. Let HL = HB U ¬.HB where for any set of ground atoms S, ¬.S = { ¬a ¦ a ∈ S }.

A partial Herbrand interpretation I is a set of ground literals satisfying the condition that there is no atom a such that both a and the complementary of a, ¬a, belong to I.

The Fitting's semantics [F] of a logic program P is defined as the least fixed point of the following operator:

T_P: 2^{HL} -> 2^{HL}

$T_P(I)$ = { a ¦ there is clause with head a in G_P whose body is true wrt I}

U { ¬a ¦ the body of every clause with head a in G_P is false wrt I}

As T_P is monotonic, T_P has a least fixed point.

Definition [F]

The Fitting's semantics of a logic program P, denoted $SEM_F(P)$, is defined as the least fixpoint of T_P. □

We often call Fitting's semantics just F-semantics, for short.

For each logic program P, an argumentation framework $AF_0(P)$ = $(AR_0, attacks_0)$ is defined as follows:

AR_0 = {(a,K) ¦ there is clause with head a and body K in G_P}

U { (¬a,{¬a}) ¦ a ∈ HB }

(k,K) attacks (h,H) if ¬k ∈ H.

If A=(k,K) is an argument then we say that k is supported by A.

An argument of the form (¬a,{¬a}) represents the situation in a dialog where an argument which is based on an assumption a is challenged by the question "Why a ?". An illustration is given in the following example.

Example

Let P consist of the clauses:

shoes-are-wet <- grass-is-wet
grass-is-wet <- rained-last-night
rained-last-night <-

The argument for the conclusion that shoes are wet is (shoes-are-wet,{grass-is-wet}), saying that shoes are wet because grass is wet. An opponent can challenge this argument by asking "Why is grass wet". This challenge is represented in our framework as the argument (¬grass-is-wet,{¬grass-is-wet}). The proponent's response to the challenge from the opponent is the argument (grass-is-wet,{rained-last-night}) saying that grass is wet because it has rained last night. This is an answer to the question "Why is grass wet ?", and is interpreted as an attack on the argument (¬grass-is-wet,{¬grass-is-wet}). As it is a matter of fact that it has been raining last night, the opponent can not do anything but concede defeat in this game, i.e. the conclusion that shoes are wet is established. □

A game-theoretical interpretation of Fitting's semantics is established in the following theorem.

Theorem 2 (Maintheorem)

A ground literal k is true wrt Fitting's semantics, i.e. $k \in SEM_F(P)$, if and only if k is supported by an acceptable argument in $AF_0(P)$.

Proof First, let define T_0: 2^{HL} -> 2^{HL} where $T_0(I) = \{ \neg a \mid$ the body of every clause with head a in G_P is false wrt I$\}$, and T': 2^{HL} -> 2^{HL} with $T'(I) = T_0(I) \cup \{ a \mid$ there is clause with head a in G_P whose body is true wrt I U $T_0(I)\}$. It is clear that for each i there is j such that $T'^i(\emptyset) \subseteq T^j(\emptyset)$. Therefore the least fixpoint of T' and T_P coincide. We prove by induction that for each ordinal number i: $k \in T'^i(\emptyset)$ iff there is an argument (k,K): $(k,K) \in F^i(\emptyset)$.
"=>" Base step: i=0 Obvious.

Inductive step:

Case 1: i is a limit ordinal. Obvious

Case 2: i = j+1.

Case 2.1: k is positive. Hence $k \in T'^i(\emptyset)$ iff there is an argument (k,K) s.t. $K \subseteq T'^j(\emptyset) \cup T_0(T'^j(\emptyset))$. We want to show now that $(k,K) \in F^i(\emptyset)$. Let A be an attack on (k,K). We want to show that A is attacked by $F^j(\emptyset)$. There are two cases:

Case 2.1.1: A = ($\neg$b,{$\neg$b}) for some atom b. Hence b $\in$ K. Then it is clear that b $\in$ $T'^{j}(\emptyset)$. Thus there is an argument (b,H) in $F^{j}(\emptyset)$. Thus (b,H) is clearly an attack on A. That means A is attacked by $F^{j}(\emptyset)$.
Case 2.1.2: A = (b,H) for some atom b. Hence $\neg$b $\in$ K. Therefore either $\neg$b $\in$ $T'^{j}(\emptyset)$ or $\neg$b $\in$ $T_0(T'^{j}(\emptyset))$. If $\neg$b $\in$ $T'^{j}(\emptyset)$ then ($\neg$b,{$\neg$b}) $\in$ $F^{j}(\emptyset)$ from the induction hypothesis. So A is an attack on ($\neg$b,{$\neg$b}). Thus, A is attacked by $F^{j}(\emptyset)$. If $\neg$b $\in$ $T_0(T'^{j}(\emptyset))$ then there is b'$\in$ H s.t. $\neg$b'$\in$ $T'^{j}(\emptyset)$. Hence there is an argument ($\neg$b',H') $\in$ $F^{j}(\emptyset)$ from induction hypothesis. Thus A is attacked by $F^{j}(\emptyset)$. Hence (k,K) is acceptable wrt $F^{j}(\emptyset)$.

Case 2.2: k is negative, i.e. k=$\neg$b for some atom b. Thus, each argument with b as head has a false body wrt $T'^{j}(\emptyset)$. From induction hypothesis, it follows that each argument with b as head is attacked by an argument in $F^{j}(\emptyset)$. So ($\neg$b,{$\neg$b}) belongs to $F^{i}(\emptyset)$.

"<=" Analogous to the other direction. □

4. Logic Programming with Well-Founded Semantics as Games

The semantics of logic programming with negation as (possibly infinite) failure is captured by its well-founded model [GRS]. The intuitive difference between well-founded semantics and Fitting's semantics lies in the fact that well-founded semantics employs a "full" closed world assumption while Fitting's semantics presents only an "approximation" of the closed world assumption. More about these two semantics as well as other semantics of logic programming can be read in [D,L1]. Following is an example to illustrate the intuitive difference between Fitting's and well-founded semantics.

Example

Let P be the program

p <- $\neg$q
q<- q

While according to Fitting's semantics of P, both p and q are undetermined, i.e. SEM_P = $\emptyset$, the well-founded semantics of P is {$\neg$q,p}, i.e. p is "true" and q is "false". □

The definition of well-founded model is based on the following notion of unfounded sets: A set S of positive ground atoms is an unfounded set of a logic program P wrt a partial interpretation I iff for each clause C in G_P whose head belongs to S, either the

body of C is false wrt I or it contains a positive literal l such that $l \in S$.

The well-founded model [GRS] of a logic program P is defined as the least fixed point of the following monotonic operator

$$V_P(I) = \neg.GUS(I) \cup \{ a \mid \text{there is a clause in } G_P \text{ whose head is a and whose body is true wrt I} \}$$

where GUS(I) is the greatest unfounded set of P wrt I and $\neg.GUS(I) = \{ \neg b \mid b \in GUS(I) \}$.

Let $K = \{\neg b_1,\ldots,\neg b_m\}$ be a set of ground negative literals. K is said to support a ground atom k wrt P, denoted by $P,K \vdash k$, if there is a sequence of ground atoms $(e_0, e_1,\ldots,e_n)$ with $e_n = k$ such that for each e_i, either $e_i \leftarrow \in G_P$ or e_i is the head of a clause $e_i \leftarrow a_1,..,a_h,\neg a_{h+1},..,\neg a_{h+r}$ in G_P such that the positive literals $a_1,..,a_h$ belong to the preceding members in the sequence and the negative literals $\neg a_{h+1},..,\neg a_{h+r}$ belong to K.

A logic program P is transformed into an argumentation framework $AF_1(P) = \langle AR_1, attacks_1 \rangle$ as follows:

$$AR_1 = \{ (k,K) \mid K \text{ is a finite set of ground negative literals s.t. } P,K \vdash k \}$$

$$\cup \{ (\neg a,\{\neg a\}) \mid a \in HB \}$$

(k,K) attacks (k',K') iff $\neg k \in K'$

If A = (k,K) is an argument then we also say that k is supported by A.

The relation between the intuition of negation as (possibly infinite) failure and the acceptability of arguments can be explained informally as follows: $\neg a$ "holds" iff each argument supporting a is defeated where an argument (a,K) is defeated iff it is attacked by an acceptable argument.

The following theorem shows the equivalence between the well-founded semantics of P and the acceptability of arguments in $AF_1(P)$.

Theorem 3 (Maintheorem)

Let P be a logic program, and WFM be the well-founded model of P. Then for each ground literal k, k ∈ WFM if and only if k is supported by an acceptable argument in $AF_1(P)$.

Proof Let define a new operator S_p: 2^{HL} -> 2^{HL}

$S_p(I) = \neg.GUS(I) \cup \{ a \mid P,K \vdash a \text{ for some } K \subseteq \neg.GUS(I) \cup I^- \}$

where $I^- = I \cap \neg.HB$. It is not difficult to see that $lfp(S_p) = lfp(V_p)$.

We prove now by induction that for each ordinal i, $k \in S_p^i(\emptyset)$ iff there is an argument (k,K) s.t. $(k,K) \in F^i(\emptyset)$.

Base step: Obvious

Inductive step: Obvious if i is a limit ordinal. Suppose now that i = j+1

"=>" Let $k \in S_p^i(\emptyset)$. Then either $k \in \neg.GUS(I)$ or $k \in \{ a \mid P,K \vdash a$ for some $K \subseteq \neg.GUS(I) \cup I^- \}$ with $I = S_p^j(\emptyset)$.

Case 1: $k \in \neg.GUS(I)$. Let $k=\neg a$. If there is no argument of the form (a,K) then it is obvious that $(\neg a,\{\neg a\}) \in F^i(\emptyset)$. Now, let (a,K) be an argument. Then there is a proof tree Tr with root a whose negative leafs are contained in K.

Property Let $I = S_p^n(\emptyset)$, and $a \in GUS(I)$. Then each proof tree of a contains a leaf $\neg b$ such that $b \in I$

Proof By induction on n.

Base step: n=1. From the definition of GUS(I), there exist no proof tree for atoms in GUS(I). So the property holds obviously.

Induction step:

Case 1: n=m+1. We prove by induction on the depth d (the number of edges of the longest path from the root to a leaf) of Tr.

Base step: d=1. Obvious.

Induction step:d = i+1. Let K be the set of children of a in Tr. Thus a<-K is a clause in G_p. There are two cases:
Case 1: There is a positive atom a' in K s.t. $a' \in GUS(I)$. Therefore, the subtree Tr' with a' as its root is a proof tree for a' whose depth is i. So from the second induction hypothesis, there is a leaf $\neg b$ in Tr' such that $b \in I$.
Case 2: K is false wrt I, i.e. there is a literal $k \in K$ s.t. $\neg k \in I$. If k is negative then there is nothing to proved. If k is

positive then the lemma follows directly from the first induction hypothesis.

Case 2: n is a limit ordinal. Similar (end of proof of property)

From the induction hypothesis and the above property, there exists an argument (b,H) in $F^j(\emptyset)$ s.t. $\neg b \in K$. That means that the argument $(\neg a,\{\neg a\})$ with $k=\neg a$ is defended by $F^j(\emptyset)$. Hence $(\neg a,\{\neg a\}) \in F^j(\emptyset)$.

Case 2: $k \in \{ a \mid P,K \vdash a$ for some $K \subseteq \neg.GUS(I) \cup I^- \}$. Thus, there is K s.t. (k,K) is an argument and $K \subseteq \neg.GUS(I) \cup I^-$. From the case 1 and from the induction hypothesis, it follows that (k,K) is defended by $F^j(\emptyset)$. So $(k,K) \in F^j(\emptyset)$.

"<=" Analogous. □

Conclusion

We have showed that logic programming can be considered as dialog game in which a proposition is "true" if it is supported by an acceptable argument. This points out that game theoretical semantics provides a natural foundation for logic programming. As logic programming is in fact a form of nonmonotonic reasoning, we can expect that our results can be extended for nonmonotonic reasoning.

Other major approaches to semantics of logic programming is the stable model semantics and the preferred extension semantics [GL,D]. Here it is interesting to ask the question as whether or not these semantics can be also viewed as dialog games. This will be the topic of our further work.

References

[BM] Barth E.M., Martens J.L. (eds) 'Argumentation: Approaches to Theory formation', SLCS Series, Vol 8, Amsterdam/John Benjamins B.V.

[C] Clark,K.L. 'Negation as Failure' in Logic and Database, Gallaire H., Minker J. (eds), Plenum, New York,1978

[D] Dung P.M. 'Negations as Hypotheses: an Abductive Foundation for Logic Programming' In Proc. of ICLP'91, MIT Press

[F] Fitting M., 'A Kripke-Kleene Semantics for Logic Programs', in J. of LP, 1985, Vol 4, 295-312

[GL] Gelfond M.,Lifschitz V. 'The stable model semantics for logic programs', Proc. of the 5th Int Conf/Sym on Logic

Programming, MIT Press, 1988

[GRS] Van Gelder A., Ross K., Schlipf J.S. 'Unfounded sets and Well-Founded Semantics for General Logic Programs' in PODS 1988

[H] Hintikka J. 'The Game of Language', D Reidel Publishing Company, Dordrecht Holland, 1983

[K] Kowalski R.A. 'Logic for problem solving', Elsevier North Holland, New York, 1979

[L] Lorenz K., 'Dialogspiele als Semantische grundlage von Logikkakuelen', Archiv fuer Mathematische Logic und grundlagenforschung, Nov 1968

[L1] Lloyd J.W. 'Foundations of logic programming', Second edition, Springer verlag, 1987

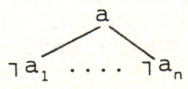
[PAA] Pereira L.M., Aparicio J.N., Alferes J.J. 'Nonmonotonic reasoning with well-founded semantics', ICLP 1991, MIT press

[P] Przymusinski T.C., 'On the Declarative Semantics of Deductive Databases and Logic Programs' in Foundations of Deductive Databases & Logic Programming, J. Minker (ed.) 1988

Appendix

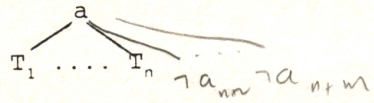
The proof trees are defined as follows:

1) If $a \leftarrow \neg a_1,\ldots,\neg a_n$ is a clause in G_p then the tree



is a proof tree of a.

2) If $a \leftarrow a_1,..,a_n,\neg a_{n+1},..,\neg a_{n+m}$ is a clause in G_p and $T_1,..,T_n$ are proof trees of $a_1,..,a_n$ repectively, then the tree



is a proof tree of a

3) There are no other proof trees.